Knuth-Bendix constraint solving is NP-complete

Konstantin Korovin and Andrei Voronkov

University of Manchester
{korovin,voronkov}@cs.man.ac.uk

Abstract. We show the NP-completeness of the existential theory of term algebras with the Knuth-Bendix order by giving a nondeterministic polynomial-time algorithm for solving Knuth-Bendix ordering constraints.

Note. The proofs in this paper are rather involved and we had to omit some of them because of the page limit. For the proofs of statements labeled with □ consult [10].

1 Introduction

Solving ordering constraints in term algebras with various reduction orders is used in rewriting to prove termination of recursive definitions, and in automated deduction to prune the search space [2, 7, 14]. Nieuwenhuis [14] connects further progress in automated deduction with constraint-based deduction.

Two kinds of orders are used in automated deduction: the Knuth-Bendix order [8] and various versions of recursive path orders [3]. The Knuth-Bendix order is used in the state-of-the-art theorem provers, for example, E [16], SPASS [17], Vampire [15], and Waldmeister [4]. There is extensive literature on solving recursive path ordering constraints (e.g., [2, 6, 13, 12]). The decidability of Knuth-Bendix ordering constraints was proved only recently in [9]. The algorithm described in that paper shows that the problem belongs to 2-NEXPTIME. It was also shown that the problem is NP-hard by reduction of the solvability of systems of linear Diophantine equations to the solvability of the Knuth-Bendix ordering constraints. In this paper we present a nondeterministic polynomial-time algorithm for solving Knuth-Bendix ordering constraints, and hence show that the problem is NP-complete. As a consequence, we obtain that the existential first-order theory of any term algebra with a Knuth-Bendix order is NP-complete too.

This paper is structured as follows. In Section 2 we define the main notions of this paper. In Section 3 we introduce the notion of isolated form of constraints and show that every constraint can be effectively transformed into an equivalent disjunction of constraints in isolated form. This transformation is represented as a nondeterministic polynomial-time algorithm computing members of this disjunction. After this, it remains to show that solvability of constraints in isolated form can be decided by a nondeterministic polynomial-time algorithm. In Section 4 we present such an algorithm using transformation to systems of linear Diophantine equations over the weights of variables. Finally, in Section 5 we complete the proof of the main result.

2 Preliminaries

A *signature* is a finite set of function symbols with associated arities. In this paper we assume an arbitrary but fixed signature Σ. *Constants* are function symbols of the arity 0. We assume that Σ contains at least one constant. We denote variables by x, y, z and terms by r, s, t. The set of all ground terms of the signature Σ can be considered as the *term algebra* of this signature, $\mathrm{TA}(\Sigma)$, by defining the interpretation $g^{\mathrm{TA}(\Sigma)}$ of any function symbol g by $g^{\mathrm{TA}(\Sigma)}(t_1, \ldots, t_n) = g(t_1, \ldots, t_n)$. For details see e.g. [5] or [11]. It is easy to see that in term algebras any ground term is interpreted by itself.

Denote the set of natural numbers by $\mathbb{N}$. We call a *weight function* on Σ any function $w : \Sigma \to \mathbb{N}$. A *precedence relation* on Σ is any linear order $\gg$ on Σ.

The definition of the Knuth-Bendix order on $\mathrm{TA}(\Sigma)$ is parametrized by (i) a weight function w on Σ; and (ii) a precedence relation $\gg$ on Σ such that (a) $w(a) > 0$ for every constant a and (b) if f is a unary function symbol and $w(f) = 0$, then f must be the greatest element of Σ w.r.t. $\gg$. These conditions on the weight function ensure that the Knuth-Bendix order is a simplification order total on ground terms (see e.g., [1]). In this paper, f will always denote a unary function symbol of the weight 0.

In the sequel we assume a fixed weight function w on Σ and a fixed precedence relation $\gg$ on Σ. We call $w(g)$ the *weight* of g. The *weight* of any ground term t, denoted $|t|$, is defined as follows: for any constant c we have $|c| = w(c)$ and for any function symbol g of a positive arity $|g(t_1, \ldots, t_n)| = w(g) + |t_1| + \ldots + |t_n|$.

The *Knuth-Bendix order* on $\mathrm{TA}(\Sigma)$ is the binary relation $\succ$ defined as follows. For any ground terms $t = g(t_1, \ldots, t_n)$ and $s = h(s_1, \ldots, s_k)$ we have $t \succ s$ if one of the following conditions holds:

1. $|t| > |s|$;
2. $|t| = |s|$ and $g \gg h$;
3. $|t| = |s|$, $g = h$ and for some $1 \leq i \leq n$ we have $t_1 = s_1, \ldots, t_{i-1} = s_{i-1}$ and $t_i \succ s_i$.

The main result of this paper is the following

Theorem 2: *The existential first-order theory of any term algebra with the Knuth-Bendix order is NP-complete.*

To prove this, we introduce a notion of Knuth-Bendix ordering constraint and show

Theorem 1: *The problem of solving Knuth-Bendix ordering constraints is NP-complete.*

The proof will be given after a series of lemmas. The idea of the proof is as follows. First, we will make $\mathrm{TA}(\Sigma)$ into a two-sorted structure by adding the sort of natural numbers, and extend its signature by (i) the weight function on ground terms; (ii) the addition function on natural numbers; (iii) the Knuth-Bendix order on ground terms.

Given an existential formula of the first-order theory of a term algebra with the Knuth-Bendix order, we will transform it step by step into an equivalent disjunction of existential formulas of the extended signature. The main aim of these steps is to replace all occurrences of $\succ$ by linear Diophantine equations on the weights of variables. After such a transformation we will obtain existential formulas consisting of linear Diophantine equations on the weight of variables plus statements expressing that, for some fixed natural number N, there exists at least N terms of the same weight as $|x|$, where x is a variable. We will then show how this statements can be expressed using systems of linear Diophantine equations on the weights of variables and use that the decidability of systems of linear Diophantine equations is in NP.

We denote by $\mathrm{TA}^+(\Sigma)$ the following structure with two sorts: the *term algebra sort* and the *arithmetical sort*. The domains of the term algebra sort and the arithmetical sort are the sets of ground terms of Σ and natural numbers, respectively. The signature of $\mathrm{TA}^+(\Sigma)$ consists (i) of all symbols of Σ interpreted as in $\mathrm{TA}(\Sigma)$, (ii) symbols $0, 1, >, +$ having their conventional interpretation over natural numbers, (iii) the binary relation symbol $\succ$ on the term algebra sort, interpreted as the Knuth-Bendix order, (iv) the unary function symbol $| \dots |$, interpreted as the weight function. When we need to distinguish the equality $=$ on the term algebra sort from the equality on the arithmetical sort, we denote the former by $=_{\mathrm{TA}}$, and the latter by $=_{\mathbb{N}}$.

We will prove the existential theory of $\mathrm{TA}^+(\Sigma)$ is in NP, from which the fact that the existential theory of any term algebra with the Knuth-Bendix order belongs to NP follows immediately. We consider *satisfiability* and *equivalence* of formulas with respect to the structure $\mathrm{TA}^+(\Sigma)$. We call a *constraint* in the language of $\mathrm{TA}^+(\Sigma)$ any conjunction of atomic formulas of this language.

Proposition 1. *The existential theory of* $\mathrm{TA}^+(\Sigma)$ *is in NP if and only if so is the constraint satisfiability problem.*

A *substitution* is a mapping from a set of variables to the set of terms. A substitution θ is called *grounding* for an expression C (i.e., term or constraint) if for every variable x occurring in C the term $\theta(x)$ is ground. Let θ be a substitution grounding for an expression C. We denote by $C\theta$ the expression obtained from C by replacing in it every variable x by $\theta(x)$. A substitution θ is called a *solution* to a constraint C if θ is grounding for C and $C\theta$ is valid.

In the sequel we will often replace a constraint $C(\bar{x})$ by a formula $A(\bar{x}, \bar{y})$ containing extra variables $\bar{y}$ and say that they are "equivalent". By this we mean that $\mathrm{TA}^+(\Sigma) \models \forall \bar{x}(C(\bar{x}) \leftrightarrow \exists \bar{y} A(\bar{x}, \bar{y}))$. In other words, the solutions to C are exactly the solutions to A projected on $\bar{x}$.

3 Isolated forms

We are interested not only in satisfiability of constraints, but also in their solutions. Our algorithm will consist of equivalence-preserving transformation steps. When the signature contains no unary function symbol of the weight 0, the transformation will preserve equivalence in the following strong sense. At each step, given a constraint $C(\bar{x})$, we transform it into constraints $C_1(\bar{x}, \bar{y}), \dots, C_n(\bar{x}, \bar{y})$ such that for every sequence of ground terms $\bar{t}$, the constraint $C(\bar{t})$ holds if and only if there exist k and a sequence of ground terms $\bar{s}$ such that $C_k(\bar{t}, \bar{s})$ holds. Moreover this transformations will be presented as nondeterministic polynomial-time algorithm which computes on every branch some $C_i(\bar{x},\bar{y})$, and every $C_i(\bar{x},\bar{y})$ is computed on at least one branch. When the signature contains a unary function symbol of the weight 0, the transformation will preserve a weaker form of equivalence: some solutions will be lost, but solvability will be preserved.

In our proof, we will reduce solvability of Knuth-Bendix ordering constraints to the problem of solvability of systems of linear Diophantine equations on the weights of variables. Condition 1 of the definition of the Knuth-Bendix order $|t| > |s|$ has a simple translation into a linear Diophantine equation, but conditions 2 and 3 do not. So we will split the Knuth-Bendix order in two partial orders: $\succ_w$ corresponding to condition 1 and $\succ_{lex}$ corresponding to conditions 2 and 3. Formally, we denote by $t \succ_w s$ the formula $|t| > |s|$ and by $t \succ_{lex} s$ the formula $|t| =_{\mathbb{N}} |s| \wedge t \succ s$. Obviously, $t_1 \succ t_2$ if and only if $t_1 \succ_{lex} t_2 \vee t_1 \succ_w t_2$. So in the sequel we will assume that $\succ$ is replaced by the new symbols $\succ_{lex}$ and $\succ_w$.

We use $x_1 \succ x_2 \succ \ldots \succ x_n$ to denote the formula $x_1 \succ x_2 \wedge x_2 \succ x_3 \wedge \ldots \wedge x_{n-1} \succ x_n$, and similar for other binary symbols in place of $\succ$.

A term t is called *flat* if t is either a variable or has the form $g(x_1,\ldots,x_m)$, where $g \in \Sigma$, $m \geq 0$, and $x_1,\ldots,x_m$ are variables. We call a constraint *chained* if (1) it has a form $t_1 \# t_2 \# \ldots \# t_n$, where each occurrence of $\#$ is $\succ_w$, $\succ_{lex}$ or $=_{\mathrm{TA}}$, (ii) each term t_i is flat, (iii) if some of the t_i's has the form $g(x_1,\ldots,x_n)$, then $x_1,\ldots,x_n$ are some of the t_j's. Denote by $\bot$ the logical constant "false".

Lemma 1. *Any constraint C is equivalent to a disjunction $C_1 \vee \ldots \vee C_k$ of chained constraints. Moreover, there exists a nondeterministic polynomial-time algorithm which, for a given C, computes on every branch either $\bot$ or some C_i; and every C_i is computed on at least one branch.* $\Box$

We will now introduce several special kinds of constraints which will be used in our proofs below, namely *arithmetical*, *triangle*, and *isolated*. A constraint is called *arithmetical* if it uses only arithmetical relations $=_{\mathbb{N}}$ and $>$, for example $|f(x)| > |a| + 3$. A constraint $y_1 =_{\mathrm{TA}} t_1 \wedge \ldots \wedge y_n =_{\mathrm{TA}} t_n$ is said to be in *triangle form* if (i) $y_1,\ldots,y_n$ are pairwise different variables, and (ii) for all $j \geq i$ the variable y_i does not occur in t_j. The variables $y_1,\ldots,y_n$ are said to be *dependent* in this constraint. A constraint is said to be *simple* if it has the form

$$x_{11} \succ_{lex} x_{12} \succ_{lex} \ldots \succ_{lex} x_{1n_1} \wedge \ldots \wedge x_{k1} \succ_{lex} x_{k2} \succ_{lex} \ldots \succ_{lex} x_{kn_k},$$

where $x_{11},\ldots,x_{kn_k}$ are pairwise different variables. A constraint is said to be in *isolated form* if either it is $\bot$ or it has the form $C_{arith} \wedge C_{triang} \wedge C_{simp}$, where C_{arith} is an arithmetical constraint, C_{triang} is in triangle form, and C_{simp} is a simple constraint such that no variable of C_{simp} is dependent in C_{triang}.

Our decision procedure for Knuth-Bendix ordering constraints is designed as follows. By Lemma 1 we can transform any constraint into an equivalent disjunction of chained constraints. Our next step is to give a transformation of any chained constraint into an equivalent disjunction of constraints in isolated form. Then in Section 4 we show how to transform any constraint in isolated form into an equivalent disjunction of systems of linear Diophantine equations on the weights of variables. Then we can use the result that the decidability of systems of linear Diophantine equations is in NP.

Let us show how to transform any chained constraint into an equivalent disjunction of isolated forms. The transformation will work on the constraints of the form

$$C_{chain} \wedge C_{arith} \wedge C_{triang} \wedge C_{simp}, \tag{1}$$

such that (i) $C_{arith}, C_{triang}, C_{simp}$ are as in the definition of isolated form; (ii) C_{chain} is a chained constraint; (iii) each variable of C_{chain} neither occurs in C_{simp} nor is dependent in C_{triang}. We will call such constraints (1) *working*. Let us call the *size* of a chained constraint C the total number of occurrences of function symbols and variables in C. Likewise, the *essential size* of a working constraint is the size of its chained part C_{chain}.

At each transformation step we will replace working constraint (1) by a disjunction of working constraints but of smaller essential sizes. Evidently, when the essential size is 0, we obtain a constraint in isolated form.

Let us prove some lemmas about solutions to constraints of the form (1). Note that any chained constraint is of the form

$$\begin{array}{c} t_{11} \# t_{12} \# \dots \# t_{1m_1} \\ \succ_w \\ \cdots \\ \succ_w \\ t_{k1} \# t_{k2} \# \dots \# t_{km_k}, \end{array} \tag{2}$$

where each $\#$ is either $=_{\mathrm{TA}}$ or $\succ_{lex}$. We call a *row* in such a constraint any maximal subsequence $t_{i1} \# t_{i2} \# \dots \# t_{im_i}$ in which $\succ_w$ does not occur. So constraint (2) contains k rows, the first one is $t_{11} \# t_{12} \# \dots \# t_{1m_1}$ and the last one $t_{k1} \# t_{k2} \# \dots \# t_{km_k}$. Note that for any solution to (2) all terms in a row have the same weight.

Lemma 2. *There exists a polynomial-time algorithm which transforms any chained constraint C into an equivalent chained constraint that is either $\perp$, or of the form (2) and has the following property. Suppose some term of the first row t_{1j} is a variable y. Then either*

1. *y has exactly one occurrence in C, namely t_{1j} itself; or*
2. *y has exactly two occurrences in C, both in the first row: some t_{1n} has the form $f(y)$ for $n < j$, and $w(f) = 0$; moreover in this case there exists at least one $\succ_{lex}$ between t_{1n} and t_{1j}.* $\square$

It is not hard to argue that the transformation of Lemma 2 does not increase the size of the constraint.

We will now take a working constraint $C_{chain} \wedge C_{arith} \wedge C_{triang} \wedge C_{simp}$, whose chained part satisfies Lemma 2 and transform it into an equivalent disjunction of working constraints of smaller essential sizes. More precisely, these constraints will be equivalent when the signature contains no unary function symbol of the weight 0. When the signature contains such a symbol f, a weaker notion of equivalence will hold.

A term s is called an *f-variant* of a term t if s can be obtained from t by a sequence of operations of the following forms: replacement of a subterm $f(r)$ by r or replacement of a subterm r by $f(r)$. Evidently, f-variant is an equivalence relation.

Two substitutions θ_1 and θ_2 are said to be f-variants if for every variable x the term $x\theta_1$ is an f-variant of $x\theta_2$. In the proof of several lemmas below we will replace a constraint $C(\bar{x})$ by a formula $A(\bar{x}, \bar{y})$ containing extra variables $\bar{y}$ and say that $C(\bar{x})$ and $A(\bar{x}, \bar{y})$ are *equivalent up to* f. By this we mean the following.

1. For every substitution θ_1 grounding for $\bar{x}$ such that $\mathrm{TA}^+(\Sigma) \models C(\bar{x})\theta_1$, there exists a substitution θ_2 grounding for $\bar{x}, \bar{y}$ such that $\mathrm{TA}^+(\Sigma) \models A(\bar{x}, \bar{y})\theta_2$, and the restriction of θ_2 to $\bar{x}$ is an f-variant of θ_1.
2. For every substitution θ_2 grounding for $\bar{x}, \bar{y}$ such that $\mathrm{TA}^+(\Sigma) \models A(\bar{x}, \bar{y})\theta_2$, there exists a substitution θ_1 such that $\mathrm{TA}^+(\Sigma) \models C(\bar{x})\theta_1$ and θ_1 is an f-variant of the restriction of θ_2 to $\bar{x}$.

Note that when the signature contains no unary function symbol of the weight 0, equivalence up to f is the same as ordinary equivalence.

Lemma 3. *Let $C = C_{chain} \wedge C_{arith} \wedge C_{triang} \wedge C_{simp}$ be a working constraint and θ_1 be a solution to C. Let θ_2 be an f-variant of θ_1 such that (i) θ_2 is a solution to C_{chain} and (ii) θ_2 coincides with θ_1 on all variables not occurring in C_{chain}. Then there exists an f-variant θ_3 of θ_2 such that (i) θ_3 is a solution to C and (ii) θ_3 coincides with θ_2 on all variables except for the dependent variables of C_{triang}.* □

This lemma will be used below in the following way. Instead of considering the set Θ_1 of all solutions to C_{chain} we can restrict ourselves to a subset Θ_2 of Θ_1 as soon as for every solution $\theta_1 \in \Theta_1$ there exists a solution $\theta_2 \in \Theta_2$ such that θ_2 is an f-variant of θ_1.

Let us call an *f-term* any term of the form $f(t)$. By the *f-height* of a term t we mean the number n such that $t = f^n(s)$ and s is not an f-term. Note that the f-terms are exactly the terms of a positive f-height. We call the *f-distance* between two terms s and t the difference between the f-height of s and f-height of t. For example, the f-distance between the terms $f(a)$ and $f(f(g(a, b))$ is -1.

Let us now prove a lemma that restricts f-height of solutions.

Lemma 4. *Let C_{chain} be a chain constraint of the form $p_l \# p_{l-1} \# \ldots \# p_1 \succ_w \ldots$, where each $\#$ is either $=_{\mathrm{TA}}$ or $\succ_{lex}$. Further, let C_{chain} satisfy the conditions of Lemma 2 and θ be a solution to C_{chain}. Then there exists an f-variant θ' of θ such that (i) θ' is a solution to C_{chain} and (ii) for every $k \in \{1, \ldots, l\}$, the f-height of $p_k\theta'$ is at most k.*

Proof. Let us first prove the following statement

(3) The row $p_l \# p_{l-1} \# \ldots \# p_1$ has a solution θ_1, such that (i) θ_1 is an f-variant of θ, (ii) for every $1 < k \leq l$ the f-distance between $p_k\theta_1$ and $p_{k-1}\theta_1$ is at most 1.

Suppose that for some k the f-distance between $p_k\theta$ and $p_{k-1}\theta$ is $d > 1$. Evidently, to prove (3) it is enough to show the following.

(4) There exists a solution θ_2 such that (i) θ_2 is an f-variant of θ, (ii) the f-distance between $p_k\theta_2$ and $p_{k-1}\theta_2$ is $d-1$, and (iii) for every $k' \neq k$ the f-distance between $p_{k'}\theta_2$ and $p_{k'-1}\theta_2$ coincides with the f-distance between $p_{k'}\theta$ and $p_{k'-1}\theta$.

Let us show (4), and hence (3). Since θ is a solution to the row, then for every $k''' \geq k$ the f-distance between any $p_{k'''}\theta$ and $p_k\theta$ is nonnegative. Likewise, for every $k'' < k-1$ the f-distance between any $p_{k-1}\theta$ and $p_{k''}\theta$ is nonnegative. Therefore, for all $k''' \geq k > k''$, the f-distance between $p_{k'''}\theta$ and $p_{k''}\theta$ is $\geq d$, and hence is at least 2. Let us prove the following.

(5) Every variable x occurring in $p_l \# p_{l-1} \# \ldots \# p_k$ does not occur in $p_{k-1} \# \ldots \# p_1$.

Let x occur in both $p_l \# p_{l-1} \# \ldots \# p_k$ and $p_{k-1} \# \ldots \# p_1$. Since the constraint satisfies Lemma 2, then $p_i = f(x)$ and $p_j = x$. Then the f-distance between $p_i\theta$ and $p_j\theta$ is 1, but by our assumption it is at least 2, so we obtain a contradiction. Hence (5) is proved.

Now note the following.

(6) If for some $k''' \geq k$ a variable x occurs in $p_{k'''}$, then $x\theta$ is an f-term.

Suppose, by contradiction, that $x\theta$ is not an f-term. Note that $p_{k'''}$ has a positive f-height, so $p_{k'''}$ is either x of $f(x)$. But we proved before that the f-distance between $p_{k'''}$ and p_{k-1} is at least 2, so x must be an f-term.

Now, to satisfy (4), define the substitution θ_2 as follows:

$$\theta_2(x) = \begin{cases} \theta(x), & \text{if } x \text{ does not occur in } p_l, \ldots, p_k, \\ t, & \text{if } x \text{ occurs in } p_l, \ldots, p_k \text{ and } \theta(x) = f(t). \end{cases}$$

By (5) and (6), θ_2 is defined correctly. We claim that θ_2 satisfies (4). The properties (i)-(iii) of (4) are straightforward by our construction, it only remains to prove that θ_2 is a solution to the row, i.e. for every k' we have $p_{k'}\theta_2 \# p_{k'-1}\theta_2$. Well, for $k' > k$ we have $p_{k'}\theta = f(p_{k'}\theta_2)$ and $p_{k'-1}\theta = f(p_{k'-1}\theta_2)$, and for $k' < k$ we have $p_{k'}\theta = p_{k'}\theta_2$ and $p_{k'-1}\theta = p_{k'-1}\theta_2$, in both cases $p_{k'}\theta_2 \# p_{k'-1}\theta_2$ follows from $p_{k'}\theta \# p_{k'-1}\theta$. The only difficult case is $k = k'$.

Assume $k = k'$. Since the f-distance between $p_k\theta$ and $p_{k-1}\theta$ is $d > 1$, we have $p_k\theta \neq p_{k-1}\theta$, and hence $p_k \# p_{k-1}$ must be $p_k \succ_{lex} p_{k-1}$. Since θ is a solution to $p_k \succ_{lex} p_{k-1}$ and since θ_2 is an f-variant of θ, the weights of $p_k\theta_2$ and $p_{k-1}\theta_2$ coincide. But then $p_k\theta_2 \succ_{lex} p_{k-1}\theta_2$ follows from the fact that the f-distance between $p_k\theta_2$ and $p_{k-1}\theta_2$ is $d-1 \geq 1$.

Now the proof of (4), and hence of (3), is completed. In the same way as (3), we can also prove

(7) The constraint C_{chain} has a solution θ' such that (i) θ' is an f-variant of θ, (ii) for every $1 < k \leq l$ the f-distance between $p_k\theta_1$ and $p_{k-1}\theta'$ is at most 1. (iii) the f-height of $p_1\theta'$ is at most 1; (iv) θ' and θ coincide on all variables occurring in the rows below the first one.

It is not hard to derive Lemma 4 from (7).

Lemma 5. *Let $C = C_{chain} \wedge C_{arith} \wedge C_{triang} \wedge C_{simp}$ be a working constraint in which C_{chain} is nonempty. There exists a nondeterministic polynomial-time algorithm which transforms C into a disjunction of working constraints having C_{chain} of smaller sizes and equivalent to C up to f.*

Proof. Consider the first row of C_{chain}. Let this row be $p_l \# p_{l-1} \# \dots \# p_1$. Then C_{chain} has the form $p_l \# p_{l-1} \# \dots \# p_1 \succ_w t_1 \# \dots \# t_n$. If $l = 1$, i.e., the first row consists of one term, we can remove this row and add $|p_1| > |t_1|$ to C_{arith} obtaining an equivalent constraint with smaller essential size. So we assume that the first row contains at least two terms.

As before, we assume that f is a unary function symbol of the weight 0. By Lemma 4, if some p_i is either a variable x or a term $f(x)$, it is enough to search for solutions θ such that the height of $x\theta$ is at most l.

A term is called *quasi-flat* if it has the form $f^k(t)$ where t is flat. We will now get rid of equalities in the first row, but by introducing quasi-flat terms instead of the flat ones. When we use notation $f^k(t)$ below, we assume $k \geq 0$, and $f^0(t)$ will stand for t. Let us first get rid of equalities of the form $f^k(x) =_{\mathrm{TA}} f^m(y)$ and then of all other equalities.

If the first row contains an equality $f^k(x) =_{\mathrm{TA}} f^m(y)$, we do the following. If x coincides with y and $k \neq m$, then the constraint is unsatisfiable. If x coincides with y and $k = m$, we replace $f^k(x) =_{\mathrm{TA}} f^m(y)$ by $f^k(x)$. Assume now that x is different from y. Without loss of generality assume $k \geq m$. We add $y =_{\mathrm{TA}} f^{k-m}(x)$ to C_{triang}, and replace other occurrences of y in C_{chain} (if any) by $f^{k-m}(x)$. Note that other occurrences of y in C_{chain} can only be in the first row, and only in the terms $f^r(y)$.

After all these transformations we can assume that equalities $f^k(x) =_{\mathrm{TA}} f^m(y)$ do not occur in the first row.

If the first row contains an equality $x =_{\mathrm{TA}} t$ between a variable and a term, we replace this equality by t, replace all occurrences of x by t in the first row, and add $x =_{\mathrm{TA}} t$ to C_{triang} obtaining an equivalent working constraint.

If the first row contains an equality $g(x_1, \dots, x_m) =_{\mathrm{TA}} h(t_1, \dots, t_n)$ where g and h are different function symbols, the constraint is unsatisfiable.

If the first row contains an equality $g(x_1, \dots, x_n) =_{\mathrm{TA}} g(y_1, \dots, y_n)$ we do the following. If the term $g(x_1, \dots, x_n)$ coincides with $g(y_1, \dots, y_n)$, replace this equality by $g(x_1, \dots, x_n)$. Otherwise, find the smallest number i such that x_i is different from y_i and (i) add $y_i =_{\mathrm{TA}} x_i$ to C_{triang}; (ii) replace all occurrences of y_i in C_{chain} by x_i.

So we can now assume that the first row contains no equalities and hence it has the form $q_n \succ_{lex} q_{n-1} \succ_{lex} \dots \succ_{lex} q_1$, where all of the terms q_i are quasi-flat.

If all of the q_i are variables, we can move $q_n \succ_{lex} q_{n-1} \succ_{lex} \dots \succ_{lex} q_1$ to C_{simp} and add $|q_1| > |t_1|$ to C_{arith} obtaining an equivalent working constraint of smaller essential size. Hence, we can assume that at least one of the q_i is a nonvariable term.

Take any term q_k in the first row such that q_k is either a variable x or a term $f^r(x)$. Note that other occurrences of x in C_{chain} can only be in the first row, and only in the terms of the form $f^k(x)$.

Consider the formula G defined as

$$\bigvee_{g \in \Sigma - \{f\}} \bigvee_{m=0\dots l} x =_{\mathrm{TA}} f^m(g(\bar{y})). \tag{8}$$

where $\bar{y}$ is a sequence of pairwise different new variables. Since we proved that it is enough to restrict ourselves to solutions θ for which the height of $x\theta$ is at most l, the formulas C and $C \wedge G$ are equivalent up to f.

Using the distributivity laws, $C \wedge G$ can be turned into an equivalent disjunction of formulas $x =_{\mathrm{TA}} f^m(g(\bar{y})) \wedge C$. For every such formula, replace x by $f^m(g(\bar{y}))$ in the first row, and add $x =_{\mathrm{TA}} f^m(g(\bar{y}))$ to the triangle part. We do this transformation for all terms in the first row of the form $f^k(z)$, where $k \geq 0$ and z is a variable.

Let us show how to replace constraints of the first row with equivalent constraints consisting of constraints on variables and arithmetical constraints. Consider the pair q_n, q_{n-1}. Now $q_n = f^k(g(x_1, \ldots, x_u))$ and $q_{n-1} = f^m(h(y_1, \ldots, y_v))$ for some variables $x_1, \ldots, x_u, y_1, \ldots, y_v$ and function symbols $g, h \in \Sigma - \{f\}$. Then $q_n \succ_{lex} q_{n-1}$ is $f^k(g(x_1, \ldots, x_u)) \succ_{lex} f^m(h(y_1, \ldots, y_v))$. If $k < m$ or ($k = m$ and $h \gg g$), then $f^k(g(x_1, \ldots, x_u)) \succ_{lex} f^m(h(y_1, \ldots, y_v))$ is equivalent to $\bot$. If $k > m$ or ($k = m$ and $g \gg h$), then $f^k(g(x_1, \ldots, x_u)) \succ_{lex} f^m(h(y_1, \ldots, y_v))$ is equivalent to the arithmetical constraint $|g(x_1, \ldots, x_u)| =_{\mathbb{N}} |h(y_1, \ldots, y_v)|$ which can be added to C_{arith}. If $k = m$ and $g = h$ (and hence $u = v$), then

$$f^k(g(x_1, \ldots, x_u)) \succ_{lex} f^m(h(y_1, \ldots, y_v)) \leftrightarrow |g(x_1, \ldots, x_u)| =_{\mathbb{N}} |h(y_1, \ldots, y_v)| \wedge \bigvee_{i=1\ldots u} (x_1 =_{\mathrm{TA}} y_1 \wedge \ldots \wedge x_{i-1} =_{\mathrm{TA}} y_{i-1} \wedge x_i \succ y_i).$$

We can now do the following. Add $|g(x_1, \ldots, x_u)| =_{\mathbb{N}} |h(y_1, \ldots, y_v)|$ to C_{arith} and replace $q_n \succ_{lex} q_{n-1}$ with the equivalent disjunction

$$\bigvee_{i=1\ldots u} (x_1 =_{\mathrm{TA}} y_1 \wedge \ldots \wedge x_{i-1} =_{\mathrm{TA}} y_{i-1} \wedge x_i \succ y_i).$$

Then using the distributivity laws turn this formula into the equivalent disjunction of constraints of the form $C \wedge x_1 =_{\mathrm{TA}} y_1 \wedge \ldots \wedge x_{i-1} =_{\mathrm{TA}} y_{i-1} \wedge x_i \succ y_i$ for all $i = 1 \ldots u$. For each of these constraints, we can move, as before, the equalities $x =_{\mathrm{TA}} y$ one by one to the triangle part C_{triang}, and make $C_{chain} \wedge x_i \succ y_i$ into a disjunction of chained constraints as in Lemma 1.

Let us analyze what we have achieved. After these transformations, in each member of the obtained disjunction the first row is removed from the chained part C_{chain} of C. Since the row contained at least one function symbol, each member of the disjunction will contain at least one occurrence of a function symbol less than the original constraint. This is enough to prove termination of our algorithm, but not enough to present it as a nondeterministic polynomial-time algorithm. The problem is that, when p_n is a variable x or a term $f(x)$, one occurrence of x in p_n can be replaced by one or more constraints of the form $x_i \succ y_i$, where x_i and y_i are new variables. To be able to show that the essential sizes of each of the resulting constraints is strictly less than the essential size of the original constraint, we have to modify our algorithm slightly.

The modification will guarantee that the number of new variables introduced in the chained part of the constraint is not more than the number of variables eliminated from the first row. We will achieve this by moving some constraints to the simple part C_{simp}.

The new variables only appear when we replace a variable in the first row by a term $f^k(h(u_1, \ldots, u_m))$ or by $f^k(h(v_1, \ldots, v_m))$ obtaining a constraint $f^k(h(u_1, \ldots, u_m)) \succ_{lex} f^k(h(v_1, \ldots, v_m))$, which is then replaced by

$$u_1 =_{\mathrm{TA}} v_1 \wedge \ldots \wedge u_{i-1} =_{\mathrm{TA}} v_{i-1} \wedge u_i \succ v_i. \qquad (9)$$

Let us call a variable u_i (respectively, v_i) *new* if $f^k(h(u_1, \ldots, u_m))$ appeared in the first row when we replaced a variable by a nonvariable term containing h using formula (8). In other words, new variables are those that did not occur in the first row before our transformation, but appeared in the first row during the transformation. All other variables are called *old*. After the transformation we obtain a conjunction E of constraints of the form $x_i =_{\mathrm{TA}} x_j$ or $x_i \succ x_j$, where x_i, x_j can be either new or old. Without loss of generality we can assume that this conjunction of constraints does not contain chains of the form $x_1 \# \ldots \# x_n \# x_1$ where $n \geq 2$ and at least one of the $\#$'s is $\succ$. Indeed, if E contains such a chain, then it is unsatisfiable.

We will now show that the number of new variables can be restricted by moving constraints on these variables into the triangle or isolated part. Among the new variables, let us distinguish the following three kinds of variables. A new variable x is called *blue in* E if E contains a chain $x =_{\mathrm{TA}} x_1 =_{\mathrm{TA}} \ldots =_{\mathrm{TA}} x_n$, where x_n is an old variable. Evidently, a blue variable x causes no harm since it can be replaced by an old variable x_n. Let us denote by $\prec$ the inverse relation to $\succ$. A new variable x is called *red in* E if it is not blue in E and E contains a chain $x \# x_1 \# \ldots \# x_n$, where x_n is an old variable, and all of the $\#$'s are either $=_{\mathrm{TA}}$, or $\succ$, or $\prec$. Red variables are troublesome, since there is no obvious way to get rid of them. However, we will show that the number of red variables is not greater than the number of replaced variables (such as the variable x in (8)). Finally, all variables that are neither blue nor red in E are called *green* in E.

Getting rid of green variables. We will now show that the green variables can be moved to the simple part of the constraint C_{simp}. To this end, note an obvious property: if E contains a constraint $x \# y$ and x is green, then y is green too. We can now do the following with the green variables. As in Lemma 1, we can turn all the green variables into a disjunction of chained constraints of the form $v_1 \# \ldots \# v_n$, where $\#$ are $=_{\mathrm{TA}}$, $\succ_w$, or $\succ_{lex}$, and use the distributivity laws to obtain chained constraints $v_1 \# \ldots \# v_n$. Let us call this constraint a *green chain*. Then, if there is any equality $v_i =_{\mathrm{TA}} v_{i+1}$ in the green chain, we add this equality to C_{triang} and replace this equality by v_{i+1} in the chain. Further, if the chain has the form $v_1 \succ_{lex} \ldots \succ_{lex} v_k \succ_w v_{k+1} \# \ldots \# v_n$, we add $v_1 \succ_{lex} \ldots \succ_{lex} v_k$ to C_{simp} and $|v_k| > |v_{k+1}|$ to C_{arith}, and replace the green chain by $v_{k+1} \# \ldots \# v_n$. We do this transformation until the green chain becomes of the form $v_1 \succ_{lex} \ldots \succ_{lex} v_k$. After this, the green chain can be removed from E and added to C_{simp}. Evidently, this transformation can be presented as a nondeterministic polynomial-time algorithm.

Getting rid of blue variables. If E contains a blue variable x, then it also contains a chain of constraints $x =_{\mathrm{TA}} x_1 =_{\mathrm{TA}} \ldots =_{\mathrm{TA}} x_n$, where x_n is an old variable. We replace x by x_n in C and add $x =_{\mathrm{TA}} x_n$ to the triangle part C_{triang}.

Red variables. Let us show the following: in every term $f^k(h(u_1, \ldots, u_m))$ in the first row at most one variable among $u_1, \ldots, u_m$ is red. It is not hard to argue that it is sufficient to prove a stronger statement: if for some i the variable u_i is red, then all variables $u_1, \ldots, u_{i-1}$ are blue. So suppose u_i is red and $u_i \# y_n \# \ldots \# y_1$ is a shortest chain in E such that y_1 is old. We prove that the variables $u_1, \ldots, u_{i-1}$ are blue, by induction on n. When $n = 1$, E contains either the constraint $u_i \succ y_1$ or $y_1 \succ u_i$, where y_1 is old. Without loss of generality assume that E contains $u_i \succ y_1$. Then (cf. (9)) this equation appeared in E when we replaced $f^k(h(u_1,\ldots,u_m)) \succ_{lex} f^k(h(v_1,\ldots,v_m))$ by $u_1 =_{\mathrm{TA}} v_1 \wedge \ldots \wedge u_{i-1} =_{\mathrm{TA}} v_{i-1} \wedge u_i \succ v_i$ and $y_1 = v_i$. But then E also contains the equations $u_1 =_{\mathrm{TA}} v_1,\ldots,u_{i-1} =_{\mathrm{TA}} v_{i-1}$, where the variables $v_1,\ldots,v_{i-1}$ are old, and so the variables $u_1,\ldots,u_{i-1}$ are blue. The proof for $n > 1$ is similar, but we use the fact that $v_1,\ldots,v_{i-1}$ are blue rather than old.

To complete the transformation, we add all constraints on red variables to C_{chain} and make C_{chain} into a disjunction of chained constraint as in Lemma 1.

When we completed the transformation on the first row, the row disappears from the chained part C_{chain} of C. If the first row contained no function symbols, the size of C_{chain} will become smaller, since several variables will be removed from it. If C_{chain} contained at least one function symbol, that after the transformation the number of occurrences of function symbols in C_{chain} will decrease. Some red variables will be introduced, but we proved that their number is not greater than the number of variables eliminated from the first row. Therefore, the size of C_{chain} strictly decreases after the transformation.

Again, it is not hard to argue that the transformation can be presented as a nondeterministic polynomial-time algorithm computing all members of the resulting disjunction of constraints.

Lemmas 1 and 5 imply the following:

Lemma 6. *Let C be a constraint. Then there exists a disjunction $C_1 \vee \ldots \vee C_n$ of constraints in isolated form equivalent to C up to f. Moreover, members of such a disjunction can be found by a nondeterministic polynomial-time algorithm.*

Our next aim is to present a nondeterministic polynomial-time algorithm solving constraints in isolated form.

4 From constraints in isolated form to systems of linear Diophantine Equations

Let C be a constraint in isolated form $C_{simp} \wedge C_{arith} \wedge C_{triang}$. Our decision algorithm will be based on a transformation of the simple constraint C_{simp} into an equivalent disjunction D of arithmetical constraints. Then we can check the satisfiability of the resulting formula $D \wedge C_{arith}$ by using an algorithm for solving systems of linear Diophantine equations on the weights of variables.

To transform C_{simp} into an arithmetical formula, observe the following. The constraint C_{simp} is a conjunction of the constraints of the form $x_1 \succ_{lex} \ldots \succ_{lex} x_N$ having no common variables. To solve such a constraint we have to ensure that at least N different terms of the same weight as x_1 exist.

In this section we will show that for each N the statement "there exists at least N different terms of a weight w" can be expressed as an existential formula of w in Presburger's Arithmetic.

Lemma 7. *There exists a polynomial time of N algorithm, which constructs an existential formula $at\_least_N(x)$ of Presburger arithmetic valid on a natural number x if and only if there exists at least N different terms of the weight x.* □

5 Main result

Now we can prove NP-completeness of the constraint solving:

Theorem 1. *Knuth-Bendix ordering constraint solving is NP-complete.*

Proof. By Proposition 1 it is enough to prove decidability of the constraint satisfaction problem. Take a constraint. By Lemma 5 it can be effectively transformed into an equivalent disjunction of isolated forms, so it remains to show how to check satisfiability of constraints in isolated form.

Suppose that C is in isolated form. Recall that C is of the form

$$C_{arith} \wedge C_{triang} \wedge C_{simp}. \tag{10}$$

Let C_{simp} contain a chain $x_1 \succ_{lex} \ldots \succ_{lex} x_N$ such that $x_1, \ldots, x_N$ does not occur in the rest of C_{simp}. Denote by C'_{simp} the constraint obtained from C_{simp} by removing this chain. It is not hard to argue that C is equivalent to the constraint

$$C_{arith} \wedge C_{triang} \wedge C'_{simp} \wedge \bigwedge_{i=2\ldots N} (|x_i| =_{\mathbb{N}} |x_1|) \wedge at\_least_N(|x_1|).$$

In this way we can replace C_{simp} by an arithmetical constraint, so we assume that C_{simp} is empty. Let C_{triang} have the form $y_1 =_{\mathrm{TA}} t_1 \wedge \ldots \wedge y_n =_{\mathrm{TA}} t_n$. Let Z be the set of all variables occurring in $C_{arith} \wedge C_{triang}$. It is not hard to argue that $C_{arith} \wedge C_{triang}$ is satisfiable if and only if the following constraint is satisfiable:

$$C_{arith} \wedge |y_1| =_{\mathbb{N}} |t_1| \wedge \ldots \wedge |y_n| =_{\mathbb{N}} |t_n| \wedge \textstyle\bigwedge_{z \in Z} at\_least_1(|z|).$$

So we reduced the decidability of the existential theory of term algebras with a Knuth-Bendix order to the problem of solvability of systems of linear Diophantine equations. Our proof can be represented as a nondeterministic polynomial-time algorithm.

Let us refer to [9] where it is shown that the problem is NP-hard by reduction of the solvability of systems of linear Diophantine equations to the solvability of the Knuth-Bendix ordering constraints.

This result and Theorem 1 implies the main result of this paper.

Theorem 2. *The existential first-order theory of any term algebra with the Knuth-Bendix order is NP-complete.*

Related work is discussed in [10].

References

1. F. Baader and T. Nipkow. *Term Rewriting and and All That*. Cambridge University press, Cambridge, 1998.
2. H. Comon. Solving symbolic ordering constraints. *International Journal of Foundations of Computer Science*, 1(4):387–411, 1990.

3. N. Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.
4. Th. Hillenbrand, A. Buch, R. Vogt, and B. Löchner. Waldmeister: High-performance equational deduction. *Journal of Automated Reasoning*, 18(2):265–270, 1997.
5. W. Hodges. *Model theory*. Cambridge University Press, 1993.
6. J.-P. Jouannaud and M. Okada. Satisfiability of systems of ordinal notations with the subterm property is decidable. In J.L. Albert, B. Monien, and M. Rodríguez-Artalejo, editors, *Automata, Languages and Programming, 18th International Colloquium, ICALP'91*, volume 510 of *Lecture Notes in Computer Science*, pages 455–468, Madrid, Spain, 1991. Springer Verlag.
7. H. Kirchner. On the use of constraints in automated deduction. In A. Podelski, editor, *Constraint Programming: Basics and Tools*, volume 910 of *Lecture Notes in Computer Science*, pages 128–146. Springer Verlag, 1995.
8. D. Knuth and P. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford, 1970.
9. K. Korovin and A. Voronkov. A decision procedure for the existential theory of term algebras with the Knuth-Bendix ordering. In *Proc. 15th Annual IEEE Symp. on Logic in Computer Science*, pages 291–302, Santa Barbara, California, June 2000.
10. K. Korovin and A. Voronkov. Knuth-Bendix constraint solving is NP-complete. Preprint CSPP-8, Department of Computer Science, University of Manchester, February 2000.
11. M. Maher. Complete axiomatizations of the algebras of finite, rational and infinite trees. In *Proc. IEEE Conference on Logic in Computer Science (LICS)*, pages 348–357, 1988.
12. P. Narendran, M. Rusinowitch, and R. Verma. RPO constraint solving is in NP. In G. Gottlob, E. Grandjean, and K. Seyr, editors, *Computer Science Logic, 12th International Workshop, CSL'98*, volume 1584 of *Lecture Notes in Computer Science*, pages 385–398. Springer Verlag, 1999.
13. R. Nieuwenhuis. Simple LPO constraint solving methods. *Information Processing Letters*, 47:65–69, 1993.
14. R. Nieuwenhuis. Rewrite-based deduction and symbolic constraints. In H. Ganzinger, editor, *Automated Deduction—CADE-16. 16th International Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence, pages 302–313, Trento, Italy, July 1999.
15. A. Riazanov and A. Voronkov. Vampire. In H. Ganzinger, editor, *Automated Deduction—CADE-16. 16th International Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence, pages 292–296, Trento, Italy, July 1999.
16. S. Schulz. System abstract: E 0.3. In H. Ganzinger, editor, *Automated Deduction—CADE-16. 16th International Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence, pages 297–301, Trento, Italy, July 1999.
17. C. Weidenbach, B. Afshordel, U. Brahm, C. Cohrs, T. Engel, E. Keen, C. Theobalt, and D. Topic. System description: SPASS version 1.0.0. In H. Ganzinger, editor, *Automated Deduction—CADE-16. 16th International Conference on Automated Deduction*, volume 1632 of *Lecture Notes in Artificial Intelligence*, pages 378–382, Trento, Italy, July 1999.